CHAPTER

13

FAULT TOLERANCE

After reading this chapter and completing the exercises, you will be able to:

- Define the parts of an operating system that make it fault-tolerant
- Explain how the various backup operations work
- List the backup media supported by Windows 2000
- Explain what RAID is and how the various forms of RAID supported by Windows 2000 contribute to its fault tolerance
- Define clustering and explain how it works

A key component of any operating system is availability. That is, you shouldn't have to wonder whether your data will be available for use. To ensure that your data remain available to users at all times, you must implement some form of **fault tolerance**.

To provide this high availability, the operating system you're using should be fault-tolerant. Fault-tolerant systems protect user and system configuration data and make sure that the computer itself remains available. In the course of this chapter, you will learn about the methods that an operating system can employ to be fault-tolerant, such as backups, fault-tolerant disk arrays, and (for Windows 2000 Advanced Server and Windows 2000 Datacenter) **clustering**.

Backing Up and Restoring Data

All computers fail sooner or later—hard disks stop working, drive controllers fail, power glitches zap motherboards, and so forth. Although you can take prophylactic measures to keep this failure from affecting production too badly, you can't prevent a computer from ever failing. All you can do is make sure that the computer is as easily replaceable as possible if it does fail.

The first line of defense for making a computer replaceable is having a current and reliable backup. If you have a reliable backup of your data and your computer stops working, you can load the data somewhere else. After all, the important part of the computer is not the computer itself, but rather the files you create and store on it. In addition to backing up the files on your computer, you should back up your system configuration data; you can then re-create the behavior of your system as well as the data it contained. You will especially want to take this step if your computer has a complicated configuration or you have done much fine-tuning to get your computer exactly the way you like it. For example, if you have backed up your system and application configuration data, you can reinstall WordCruncher on the new machine and then restore all of the special settings you made for this program, such as file locations, locations and contents of toolbars, new templates, and the like. You can always rebuild application configuration information, but it is a long and tedious process.

Backing up system configuration data—called **system state data** in Windows 2000—is even more important than backing up application configuration data. This importance stems from the fact that some data structures are identified to the operating system not by the names you give them, but by their **security identification (SID) numbers**. You cannot manually edit these SID numbers from the Windows 2000 tools. For example, imagine that you have a domain called ALPHA that has a single domain controller named ROVER. One day, the domain controller fails. You buy a new computer, reinstall Windows 2000, create a domain named ALPHA, and name the domain controller ROVER. You may think everything is fixed, but it is not. No one will be able to log onto the domain, because the domain to which everyone else belonged had a different SID number than the one you just created. To remedy this problem, all the computers must rejoin the ALPHA domain.

On a similar note, imagine that you configured security auditing for certain people to allow them to monitor their network activity. If you have to rebuild the account database (a long and tedious process, because you must re-create all user rights associated with those accounts), you also need to reconfigure the audit trail. To Windows 2000, the rebuilt accounts look like new accounts because their SID numbers are different from the ones that auditing was previously set up to configure. Windows 2000 doesn't pay attention to the names that you assign users, groups, or domains; it pays attention to the SID numbers.

System state data for a Windows 2000 Server computer include all of the following:

- The system Registry
- The COM+ class registration database of file associations
- The boot files for the computer
- The certificate services database (for a certificate server)
- The Active Directory structure (for a domain controller)
- The SYSVOL directory (for a domain controller)

You can't pick and choose certain parts of the system state data to back up—backing up is an all-or-nothing proposition in this case. System state data tend to be large (hundreds of megabytes, at least, and perhaps more depending on the complexity of the Active Directory structure)—keep that fact in mind when you choose a backup destination (for example, on another computer's hard drive or a tape backup).

Unless you want to take the chance that you will have to rebuild your servers and domain controllers from the ground up when computers fail, you need to back up. Read on to learn about the tools that Windows 2000 provides for this endeavor and the ways in which you can use them to easily and effectively protect your data.

Backup Tools

The mechanics behind a backup are simple. During the backup process, the backup application reads the directories and files you select and decides which files are eligible for backup by using the criteria you established when you set up the backup operation. Backing up is simply a file copy operation like the one you do when you use the COPY or XCOPY command. Files are copied from a source folder to a destination folder. To the backup tool, the source folder consists of the folder where you normally store the files, and the destination folder consists of the backup media. Restoring data from backup media works in the same way, only in reverse.

Windows 2000 comes with two tools that you can use to back up and restore user data and system configuration information: the graphical backup tool located in the System Tools section of the Accessories program group (see Figure 13-1), and a command-line tool called **NTBACKUP**. The two are similar in function, but not identical.

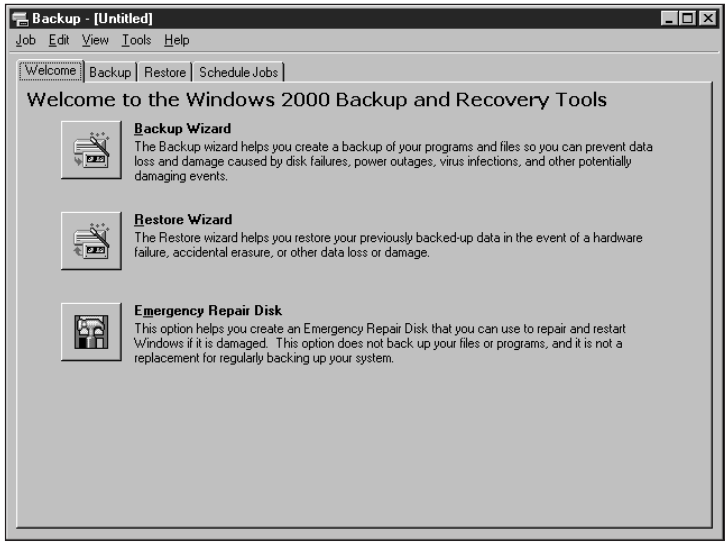

Figure 13-1 Backup tool in Windows 2000

Whichever tool you use, in the course of setting up a backup operation, you will configure some or all of the following options:

- *Backup or restore?* The backup tools can both back up and restore data.
- *Copy system state data?* Specifies that all system state data should be backed up and sets the backup type to normal or copy. This option is available only if you are backing up the local computer—for security reasons, you can't back up system configuration information over the network.
- *Name of backup file?* Backups are stored in a special file called a **selection information file** that has a .bks extension. More than one backup job can go into the same selection information file. You must create this file from the GUI before referencing it from the command line.
- *Name of the backup job?* You must assign a name to the backup job. NTBACKUP stores this job in a selection information file with a .bks extension.
- *Where will the selection information file be stored?* You can identify the backup location for each job you schedule. This backup location may be a media pool (a logical grouping of removable media, such as a tape library), a tape, a removable drive, a floppy disk, or a file on the hard disk.
- *Description of the backup job?* In addition to naming a backup job for identification purposes, you can give it a description to further delineate it.
- *Append or replace?* When backing up to a specific physical location (as opposed to backing up to a media pool), you can choose to either append the new backup to the existing data or wipe the old data out and write only the new data.

- *Verify the backup job?* Verifying a backup—that is, making sure that the final product of the backup matches the data on the original media—adds some time to the backup process but is a good way to ensure that the data were written correctly.
- *Who gets to use the backup media?* You can specify whether anyone can restore the backup or whether that privilege is restricted to only members of the Administrators group and the person who created the backup. Limiting the number of people authorized to restore a backup may make it a little more inconvenient to restore data, but it prevents unauthorized people from stealing information from backup tapes.
- *What kind of backup logging do you want to do?* A backup log is a text file that shows the process of the backup and lists any errors that occurred during the backup. The Windows 2000 backup applications support three options for backup logging: full (lists every file copied to the backup media), summary (logs only errors and important events such as the start and end of the backup), and none (which doesn't log errors at all). Generally, the summary option—the default—is the best choice because it provides you with the information you need to know. Use a full log only if you need a record of every file that was included in the backup, perhaps for auditing purposes.
- *What type of backup should be performed?* As discussed in the next section, the Windows 2000 backup applications support several backup types: normal, differential, incremental, copy, and daily. A particular backup job can perform only one of these types of backups.
- *Back up removable storage database?* Windows 2000 supports hierarchical storage management, which is a system of moving rarely used files to removable storage but maintaining a pointer to these files on the hard disk. When you access a file that's been moved, it appears as if you are opening a file stored on disk; in reality, the file is opened from tape. If you don't back up the removable storage database that records the locations of archived files, you cannot retrieve moved files because no record exists of where the files were stored.
- *Use hardware compression?* If you're backing up to a tape drive that supports hardware compression (most do), you have the option of using this compression. Go ahead and use hardware compression if it's available, because it allows you to get more use out of your tapes.

Backup Schemes

Windows 2000 supports several types of backup schemes that you can combine to back up your data quickly and thoroughly. Most backup types are dependent on a file attribute called the **archive bit** that is turned on (set) when a file is edited (or added—all new files carry the archive bit) and turned off when the backup program turns it off (resets it). Depending on the backup scheme employed by the backup program, the archive bit signals whether a file should be backed up.

Normal Backups

A **normal backup** copies every selected file to the backup media and resets the archive bit to show that the files have been backed up. Normal backups are the cornerstone of any backup program, because they are the only backup method that ensures that you get a complete copy of your file set. A network administrator's backup scheme typically calls for a normal backup at least once per week.

Backing up once per week isn't enough, however. Think about all the work you did over the past seven days. If your computer crashed, could you re-create all of it? How about those crucial insights that aren't written anywhere else, or the complex CAD drawings you did? If you had hard copies, you might be able to re-create the data, but it would be a laborious process and waste a great deal of time. Think of how long it would take an entire company to re-create a week's worth of data—while probably needing to use those data simultaneously—and you can see that backing up once weekly is not enough. Performing a normal backup every night would be another option, but backing up gigabytes of files takes a long time and requires gigabytes of storage space. Although tape storage is less expensive than disk storage, storing a complete backup for every night would add up—both in terms of buying tapes and in terms of finding storage space for those tapes. In addition, you would face the problem of finding the time to back up gigabytes of data every day. Because the Windows 2000 backup programs do not back up open files, you need a decent chunk of time when no one is using any data files to perform the backup. That may not be possible every night, especially when more people are working at home on at least a part-time basis, and communicating with the network via a dial-up connection or terminal services. Backups need to be complete, but they also need to happen as quickly as possible.

To make it easier for companies to perform fast and complete backups, the Windows 2000 backup programs include two kinds of partial backups (incremental and differential backups) that copy only a certain set of files to the backup media. You can use these types of backups to supplement your normal backups, perhaps running an incremental or daily backup every day and a normal backup once each week.

Incremental Backups

An **incremental backup** copies every file that has the archive bit set (meaning that the file is new or has been edited since the last full backup) and resets the archive bit to show that the file has been backed up. If you perform an incremental backup every day, you have a day-to-day record of which files were changed. In addition, backing up takes only as long as copying the changed files, rather than copying every file.

To restore incremental backups to a server, you must collect all of the incremental backups performed since the previous normal backup and restore them. This process ensures that the most recent files appear on the server. Incremental backups take little time to perform because the collection of the files that change each day (or however often you back up) is bound to be small. Restoring the files may take some time, however, because you must restore each backup set separately. On the other hand, incremental backups make it easy for you to preserve multiple copies of files, in case you need to restore a version that is not

necessarily the most recent. For example, if a virus was introduced to the system on Wednesday, you would want to back up from files saved before then.

Differential Backups

A **differential backup** copies every file that has the archive bit set, but does not reset the archive bit. Therefore, each time you back up, the backup set becomes larger because the previously backed up files are included in the backup set even if they haven't changed. To restore a server, you simply restore the last normal backup, then restore the most recent differential backup.

Although differential backups allow you to restore a server with ease, they are apt to take longer to perform than incremental backups, because you must first run the last normal backup, then perform the differential backup. They're also a little less flexible when it comes to finding a specific version of a file, because you can't easily tell which files changed on a specific day.

Creating a Backup Scheme

Most often, you won't use a single backup type. Instead, you will combine several types in some way to create a comprehensive backup plan that guarantees the following:

- A complete and updated set of data for the server
- A previous copy of data, in case the current backups are corrupted
- Maximum availability for the network—meaning that the backup system itself cannot interfere with the operation of the network

A backup can interfere with the network in several ways. As mentioned earlier, the Windows 2000 backup tools don't let you copy open files; therefore, to back up a server, you must shut down all server connections to ensure that all files are closed. Also, if you are backing up over the main network, the stream of data uses up a lot of available bandwidth. For this reason, many people implementing network backup solutions create a dedicated backup network that does not interfere with the main network.

A core part of the backup plan is the backup schedule. Microsoft suggests two possible schedule types: **Grandfather/Father/Son (GFS)** and the **Tower of Hanoi (ToH)**, named after a math puzzle.

In the GFS backup scheme, the "grandfather" is a monthly full backup, the "father" is a weekly full backup, and the "son" is a daily incremental or differential backup. GFS uses a total of 12 tapes (depending on the capacity of your tapes) or other backup media to keep a three-month record of the backup data—long enough to restore even rarely used files that someone accidentally erased two months ago. You could, of course, keep two series of the GFS scheme to create an even larger archive.

ToH is more complicated than GFS, but keeps a longer record of server data—32 weeks instead of 12 weeks. In ToH, you use five tapes labeled A–E in the following order: A B A C A B A D A B A C A B A E, and you perform a full backup on each tape. Tape A is reused every 2 weeks, tape B every 4 weeks, tape C every 8 weeks, tape D every 16 weeks, and tape E every

32 weeks. Because all of these operations are normal backups, it is a good idea to supplement ToH with a daily incremental or differential backup.

Table 13-1 shows a simple backup schedule based on the GFS backup scheme, having started with full backup tape 1 and assuming that no one works weekends.

Table 13-1 A sample backup schedule

Week	Day	Backup Type and Tape Used
Week 1	Monday	Differential tape A
	Tuesday	Differential tape B
	Wednesday	Differential tape C
	Thursday	Differential tape D
	Friday	Full backup tape 2
Week 2	Monday	Differential tape E
	Tuesday	Differential tape F
	Wednesday	Differential tape G
	Thursday	Differential tape H
	Friday	Full backup tape 3
Week 3	Monday	Differential tape A (overwrite)
	Tuesday	Differential tape B (overwrite)
	Wednesday	Differential tape C (overwrite)
	Thursday	Differential tape D (overwrite)
	Friday	Full backup tape 4
Week 4	Monday	Differential tape E (overwrite)
	Tuesday	Differential tape F overwrite)
	Wednesday	Differential tape G (overwrite)
	Thursday	Differential tape H (overwrite)
	Friday	Full backup tape 5

Other Backup Types

The Windows 2000 backup applications also include some other backup types (copy and daily backups) that are not really intended for backups, but rather more for file archiving or copying currently used files to other media. A **copy backup** works like a normal backup in that it copies all selected files to the backup media, regardless of whether the archive bit is set. It does not reset the archive bit, however. Copy backups are a good way to make a complete copy of data without interfering with the regular backup schedule—for example, if you want to make a copy of the server's data.

A **daily backup** is intended more for the itinerant worker. If you are going on a trip and need to take along the files on which you are currently working, a daily backup provides a quick and simple way of copying all of the files that were changed recently. This type of backup copies only those files that have changed on the day of the backup.

On the other hand, if you need files that were modified before the day of the backup, the daily backup cannot help you.

A Quick Tour of Backup Media Supported in Windows 2000

Unlike the Windows NT backup programs, which worked with tape drives only, the Windows 2000 backup applications recognize any form of storage. You can back up files to floppy disks, to removable drives, to a network drive, to a CD-R device, or even to the local computer's hard disk. The only requirements are that the media be compatible with Windows 2000 and have a name (either a drive letter or a path to which to mount). To verify that the media are compatible with Windows 2000, check the Microsoft Hardware Compatibility List (HCL), which includes all devices supported by the various Microsoft operating systems. The most recent HCL can be found at *www.microsoft.com/hwtest/hcl*.

Using Fault-Tolerant Disk Configurations

Backups are an important part of fault tolerance, but they have one problem: they're useless until you restore them, and restoring files takes time. This time lag might be acceptable if you are restoring a client workstation (because it inconveniences only one person). If a server goes down, however, you need to get it back up and running as quickly as possible so that everyone who depends on that server can go to work. Even better, the server should *remain* up and running so that no interruption of service occurs.

One way to ensure that a server remains running is to use a fault-tolerant disk configuration. Although disks are not the only parts of a computer that can fail, they are nevertheless the parts that are most likely to do so. Unlike many of the other computer components, disks have moving parts. Any device with moving parts is more prone to failure than a solid-state device, because the former is at risk for mechanical failures as well as electrical ones. If you power-protect memory and keep the inside of the computer cool, the disk should keep working; unfortunately, however, hard disks can fail even if they're power-protected. Hard disks are extremely vulnerable to contaminants such as smoke, pollen, hair, and dust. Consequently, it is in your best interest to make sure that the failure of a single disk does not affect your operations.

To accomplish this goal, you can group disks logically so that Windows 2000 sees the set as a single unit of storage and maintains parity information for the data stored on the **group**. If one disk in the logical group fails, another disk can take over for the one that failed—even if it died in the middle of a read or write operation. The technique of using multiple disks in this way is called **Redundant Array of Independent Disks (RAID)**. It is based on the idea that the more disks you have, the less likely it is that all of your disk space will fail at once—that's where the "redundancy" part arises.

The process of communicating with fault-tolerant disk volumes (or multidisk volumes that are not fault-tolerant) is a little different from that of communicating with simple disk volumes on basic disks. To a point, the process remains as we described in Chapter 2:

1. An application accepts an open file request from a user.
2. The application passes the request to the environmental subsystem.
3. The environmental subsystem passes the request to the file system driver.
4. The file system driver passes the read request to the disk management subsystem.

This point is where things change. The two kinds of requests now branch off. Disk read (or write) requests sent to logical drives or partitions on basic disks are handled by the fault-tolerant disk driver, which is used in Windows 2000 to communicate with basic disks and in Windows NT to communicate with fault-tolerant volumes. Disk access requests going to any volumes on a dynamic disk are handled by the logical disk manager, a driver that is unique to Windows 2000. The practical upshot of this setup is that fault-tolerant volumes in Windows 2000, which can be located only on dynamic disks, are only locally available to Windows 2000 because other operating systems don't have logical disk managers.

Fault-tolerant volumes are unavailable to locally installed operating systems other than Windows 2000. If these operating systems are accessing a volume on the dynamic disk from across the network, then no problem arises. Network access to a hard disk goes through the network card driver, which can then pass the connection to the file system drivers.

RAID Levels Supported in Windows 2000

A number of different kinds of RAID exists that combine physical disks in various ways to provide data security. Until a few years ago, Windows operating systems needed to support RAID with hardware disk arrays. The introduction of Windows NT incorporated software support into the base operating system. Windows NT—and now Windows 2000—support the RAID 1 and RAID 5 fault-tolerant disk configurations.

Disk Mirroring

RAID 1's full name is **disk mirroring**. In disk mirroring, partitions of the same size and on different physical disks are logically linked into a group called a **mirror set**. The operating system treats the mirror set as a single drive, and it is identified by a drive letter (or mounted to an empty path on a NTFS drive, as discussed in Chapter 7). An exact copy of the data is kept on both halves of the mirror set; therefore, if a disk in the set stops working, the data remain accessible. The data are not considered fault-tolerant until you replace the failed disk and remake the mirror set, but you can nevertheless read and write to the logical drive defined by the mirror set.

As an example, consider the following setup: A mirror set consisting of identical partitions on physical disks 0 and 1 has the drive letter H. When you save a document to drive H, the Windows 2000 disk controller writes the document to the partition on disk 0 and the partition on disk 1, as shown in Figure 13-2. When you open that document again, the disk controller

pulls its data from both disks, shortening the time required to transfer the document into memory. If disk 0 fails, drive H will still be available, even though the drive controller will now write data only to the partition on disk 1.

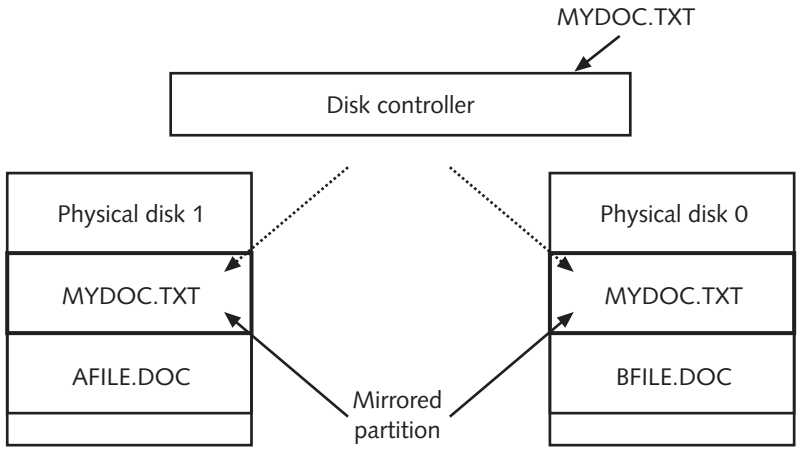

Figure 13-2 How disk mirroring works

When you mirror partitions on two disks attached to different disk controllers, you are using **disk duplexing**. Disk duplexing is even more secure than disk mirroring, because it protects you from controller failures as well as disk failures.

You can create a mirror set at the time that you create a logical partition, or you can mirror an existing partition onto an unallocated piece of space on another disk. When you create the mirror set from an existing partition, any data already on the original partition will be regenerated to the new partition in the mirror set. This strategy is therefore fault-tolerant for all data, not just data stored since you established the mirror set.

Because it requires only two physical disks, disk mirroring is an inexpensive way to make disks fault-tolerant; mirrored disks are very inefficient, however, in terms of storage space. Only half the space in a mirror set is available for storing original data—the other half is reserved for an exact copy of the data. That's a lot of overhead. No matter how large the mirror set, the proportion of overhead to actual data always remains 1:1. For this reason, most people prefer to use the other form of fault-tolerant RAID supported by Windows 2000: RAID 5.

Disk Striping

RAID 5's full name is **disk striping with parity**. Like mirror sets, stripe sets are composed of partitions of the same size and located on different physical disks, yet treated by Windows 2000 as a single logical disk. When you write data to the stripe set, the data are distributed over the partitions in the stripe set in stripes, even if the partition on the first disk has more than enough room. For example, you save Myfile.doc to a stripe set. The beginning of Myfile.doc is written

to stripe 1 of member 1, more data are recorded on stripe 2 of member 2, and the rest appears on stripe 3 of member 3.

Windows 2000 supports RAID 0, a type of disk striping that works in this way. RAID 0 volumes are not fault-tolerant, however. In fact, RAID 0 is actually more prone to failure than saving data to a single disk. The reason is simple: The more disks you have, the more likely that one of those disks will stop working at any given time. If Myfile.doc is stored on three disks, one of which stops working, you cannot load Myfile.doc again, because part of its contents are no longer available. This weakness doesn't make disk striping useless by any means. In fact, it improves disk access time by allowing a disk controller to pull data from several disks at once. It does mean, however, that backing up disk stripes regularly is crucial.

The fault-tolerant form of disk striping is disk striping with parity (RAID 5). Disk striping with parity works much like disk striping, except that in addition to writing data across the partitions in the stripe set, it writes parity information for that data. This parity information is not a copy of the original data, but instead contains the instructions that allow Windows 2000 to reconstruct the data if one of the disks in the **RAID 5 volume** stops working. Figure 13-3 illustrates the distribution of data in a RAID 5 volume.

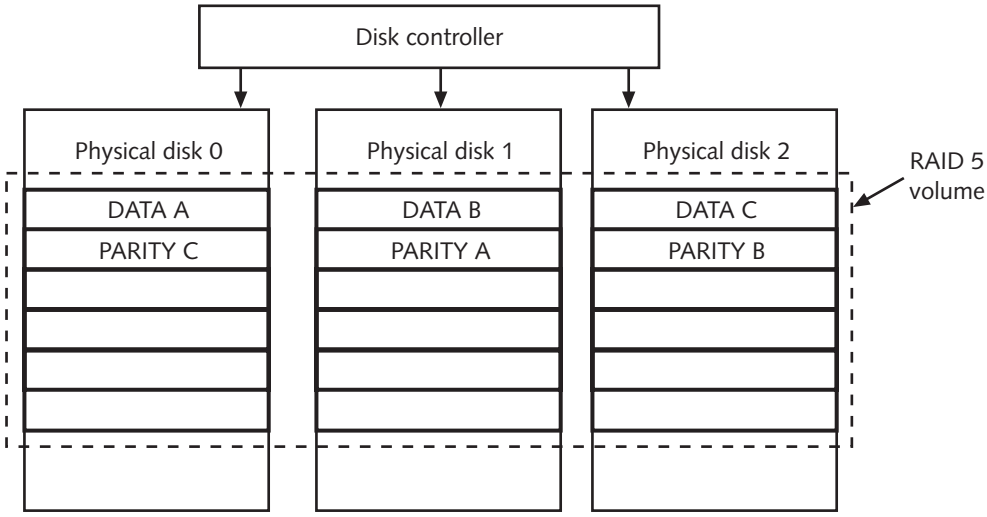

Figure 13-3 How RAID 5 volumes work

The proportion of parity data relative to user data in a RAID 5 volume is given by $1/n$, where n is the number of physical disks supporting the volume. Therefore, the more disks included in a RAID 5 volume, the more space-efficient the volume is. Sadly, you cannot add more disks to a RAID 5 volume after you have created it, nor can you make the partitions supporting it any larger. The created size is the size it will stay.

Each time that you save a file to the RAID 5 volume, the parity information for that file must be updated to reflect its current status (new time stamp, new data, or some other change). If this information is not updated regularly, the RAID 5 volume maintains backup

parity information for every file on the volume, which is not only a big waste of space, but also a security breach. When you delete a file from a volume, you want it to disappear—not to be reconstructible from parity information left behind. (If you really need the file, you should drag out the backups discussed earlier in this chapter. You still don't want someone to be able to reconstruct files that you intended to remove from the volume.)

A RAID 5 volume can update the parity information in one of two ways, both of which are dependent on an XOR calculation. On a very basic level, **exclusive OR (XOR) arithmetic** is a function that compares two 1-bit inputs and produces a single-bit output that shows the result of the comparison. If the two bits are the same, the output is 0. If the two bits are different, the output is 1. Therefore:

```
0 XOR 0 = 0
1 XOR 0 = 1
0 XOR 1 = 1
1 XOR 1 = 0
```

When you perform an XOR calculation on two numbers with more than one bit, just match the bits and calculate them individually. For example, 1101010 XOR 0101000 equals 1000010. (For numbers that are not the same number of bits, you just add more 0s to the left side of the number. For example, 1101010 and 0001101010 are the same number.) The result produced by function is the parity information, from which the original data can be recalculated by applying the parity information to whatever data remain. If you know that the result of 1101010 XOR something is 1000010, you can easily figure out which collection of 1s and 0s would produce that XOR.

One way to calculate the new parity information would be to recalculate the XOR to obtain the parity information each time that data are written to the RAID 5 volume. This approach would obviously be slow, because the disk controller would have to read the entire contents of the RAID 5 volume and perform the algorithm. A better approach—and the one that Windows 2000 uses—is to read the old data that are supposed to be overwritten and XOR it with the new data to determine the differences between the two. This calculation produces a **bit mask** that has a 1 in the position of every changed bit. (Remember that a 1 means that the two bits don't match.) After Windows 2000 has this bit mask, it can XOR it with the old parity information to compare those two numbers. This calculation yields the new parity information for the file. Although the process probably seems a little convoluted, it requires only two file reads and two XOR calculations on a small amount of data—rather than a single large file read and an XOR calculation on all data on the volume. It is much faster and less prone to error because you are dealing with a relatively small amount of data.

Faster or not, recalculating the XOR every time you change a file in a RAID 5 volume partly explains why RAID 5 volumes take longer to read data than mirrored volumes, which maintain an exact copy of a volume's data on a second volume. Although RAID 5 volumes are much more space-efficient than mirrored volumes, because of the time it takes to read the data, they are not the best choice in an environment where speed and CPU cycles are crucial. Terminal servers, in particular, should not be made fault-tolerant with RAID 5 volumes.

Creating Fault-Tolerant Volumes with the Disk Management Tool

To create fault-tolerant volumes, you need to use the Disk Management tool introduced in Chapter 7. Creating a fault-tolerant volume works like creating any other volume on a dynamic disk, with the following caveats:

- To build a mirror set, you must have two areas of unallocated space on two separate physical disks available.
- You can also mirror an existing simple volume on a dynamic disk. Other types of volumes (and volumes on basic disks) cannot be mirrored.
- To build a RAID 5 volume, you must have at least 3 and a maximum of 32 areas of unallocated space on the same number of physical disks.

The Hands-on Projects at the end of this chapter walk you through the mechanics of creating fault-tolerant volumes.

If you do not want to maintain redundant information, or if you need to mirror the data in a different place (perhaps because the other half of the mirror set has stopped working), you need to get rid of the mirror set. How you discard it depends largely on what you are trying to do. You have three options:

- Discard the data in the mirror set and remove the volume (**deleting the mirror set**).
- Keep the data in half of the mirror set (either half) but make it no longer fault-tolerant (**removing the mirror set**).
- Keep both sets of the data but make the halves independent so that you can modify the data on one without affecting the other (**breaking the mirror set**).

To destroy all data in a mirror set and convert the volume back into unallocated space, right-click the mirrored volume and choose Delete Volume from the resulting menu. The Disk Management tool will ask you to confirm your choice. Click Yes to continue deleting the mirror. Only do so if you have backed up the mirror set, because deleting the mirror set destroys the data that it contained.

A more likely scenario is that one of the disks supporting the mirror set fails. The data on the remaining disk will still be available, but no longer fault-tolerant. When this situation arises, the mirror set appears in the Disk Management tool with a Failed note on it.

To protect the data again, you need to remirror the volume. You cannot, however, remirror a volume—even if one of the disks in the mirror set has failed, the mirror set remains valid. Therefore, you need to remove the mirror set. Right-click the failed mirror set, and choose Remove Mirror from the resulting menu. You will see a dialog box showing the two halves of the mirror. Pick the half you don't want to keep and click Remove Mirror. When asked to confirm your choice, click Yes to continue. The half of the mirror set you selected returns to unallocated space, and the remaining half of the mirror set becomes a simple volume.

If both halves of the disk are still working but you don't want to mirror the data anymore, you can break the mirror set, thereby making the two volumes act like simple volumes again.

Right-click a mirror set that is still functioning (if the mirrored volume has failed, you have to remove the mirror, not break it), and choose Break Mirror from the resulting menu. You will see a message asking you to confirm your choice and warning you that your data will no longer be fault-tolerant. Click Yes to continue. Once broken, the two halves of the mirrored volume become separate simple volumes. One half retains the drive letter that had belonged to the mirrored volume; the other half gets the next available drive letter.

SERVER CLUSTERING

Server clustering takes logical disk grouping a step further. Rather than using disk arrays to provide fault tolerance and improve performance, clustering employs server arrays of two or more computers to achieve this goal. More specifically, clustering is designed to:

- Improve resource availability by keeping resources available even in case of hardware or software failure
- Improve system scalability by enabling you to add more resources to the network without making it obvious that you are doing so
- Improve server manageability by making it possible to administer a collection of servers from a single interface

Clustering Terminology

Before getting into a detailed discussion of how clustering in Windows 2000 Advanced Server and Windows 2000 Datacenter works (Windows 2000 Server does not support clustering on its own), you need to understand the terminology involved.

How Clusters Perceive Servers and Applications

The servers in a cluster are individually known as **nodes**. From the client's perspective, each node in a cluster looks like a single server, called a **virtual server**. Each virtual server is identified by a single network name and IP address (see Figure 13-4). That is, when a client machine addresses a cluster, the client initiates a connection to BIGCLUSTER, not to Node Alpha of BIGCLUSTER. The node in the cluster that actually takes over the connection depends on which server is available for client requests. To the client, however, it does not matter whether Alpha or Beta takes the request. Similarly, when a client accesses a cluster resource, that resource is identified as part of the virtual server, rather than being associated with a particular node.

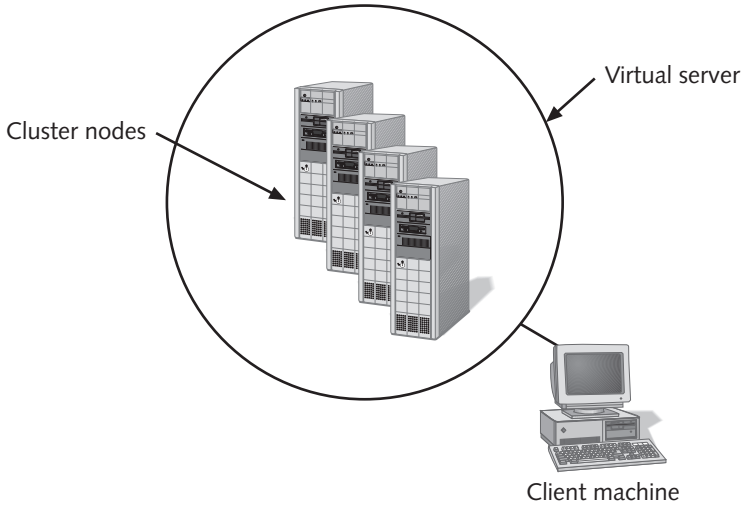

Figure 13-4 How clients perceive cluster nodes

The **cluster service** running on these nodes manages all cluster-specific activity, such as replication of Registry information between nodes and failover. The hardware and software that the cluster service manages are collectively known as **resources**. These resources can include devices such as network cards or physical disks, or logical items such as IP addresses, logical disk volumes, applications, and databases. When a resource provides its service on its node, it is considered to be **online**. A collection of resources of the same type, such as the physical disks in a cluster, is by the cluster service as a group and as a resource type. A resource can be owned by only a single node at one time, although the cluster service can handle multiple resources of the same type if the resource itself may be duplicated.

One special kind of resource, called a **quorum resource**, is used as a tiebreaker when a cluster is being booted. The node in the cluster that controls the quorum resource (often a physical disk, though it doesn't have to be) forms the cluster, and every other node in the cluster must follow its lead. The node does not necessarily keep the quorum resource while the cluster is online; if necessary, the resource can be moved to another node during failover or for some other reason.

The cluster service manages resources as a collection of dynamic link libraries (DLLs). The Windows 2000 clustering product includes DLLs for the following resource types:

- Physical disks
- Logical volumes consisting of one or more physical disks
- File and print shares
- Network addresses and names
- Services and applications (a generic DLL)
- Internet server service (a DLL specific to IIS)

Microsoft also provides a software developer's kit (SDK) that developers can use to create more DLLs for resources that should fail over from one node to another. For example, imagine that you want to cluster database servers and make sure that a specific database remains available. You could use the generic application DLL to define the database application as a resource. If you did so, however, the entire database application (and all databases associated with that application) would have to fail over to the secondary server at failover time. The more data you have to fail over, the longer it takes. Consequently, this scheme is not the best way to run things. If you defined a resource for a specific database, you could fail over just that database, without having to move everything.

Parts of the Cluster Service

The cluster service in Windows 2000 (Clussvc.exe) that runs on each node consists of a service such as the Internet Information Server or Terminal Service. It includes the following components:

- *Node Manager* Keeps track of cluster membership (that is, which nodes are part of the cluster) and monitors the health of the cluster.
- *Configuration Database Manager* Maintains the cluster configuration database that shows how resources are organized.
- *Resource Manager/Failover Manager* Is responsible for all management decisions pertaining to resources and groups and initiates failover to move those groups to the node where they should be in case of failover.
- *Event Processor* Holds together the components of the cluster service, handling common operations and initializing the cluster service.
- *Communications Manager* Handles communication between nodes of the cluster, via remote procedure calls (RPCs).
- *Global Update Manager* Updates cluster-wide information about the status of the cluster service components running on each node.
- *Time Service* Keeps the nodes in the cluster synchronized so that they can communicate. The time service is actually a resource, rather than part of the cluster service.
- *Resource Monitor* Supports the cluster service, but is not part of it. Instead, it runs in a separate process from the cluster service and monitors the status of each resource in the cluster.

That is fault tolerance in a nutshell. As you have seen, fault tolerance means keeping both your data and your network resources available.

Cluster Types

In terms of function, three main types of clusters exist:

- Active/active
- Active/standby
- Fault-tolerant

All types of clusters are fault-tolerant to some degree, by virtue of providing redundancy. The degree to which they are fault-tolerant and the speed with which one node can take over for another, however, depends on the type of cluster involved.

To keep the descriptions simple, the clusters in our examples have only two nodes: a primary node and a secondary node. You can, however, have more than two nodes in a cluster with some clustering packages, and you may want to add more servers to a cluster as time goes by to improve cluster responsiveness.

In an **active/active cluster** (used by Windows 2000 Advanced Server and Windows 2000 Datacenter), all of the nodes in the cluster are functioning and supporting user requests all the time. If one node fails, then it fails over to the other node, sending its workload there. The failover time can take as long as 90 seconds. At the end of the failover, the remaining server is supporting all client requests. Figure 13-5 shows how this type of clustering works.

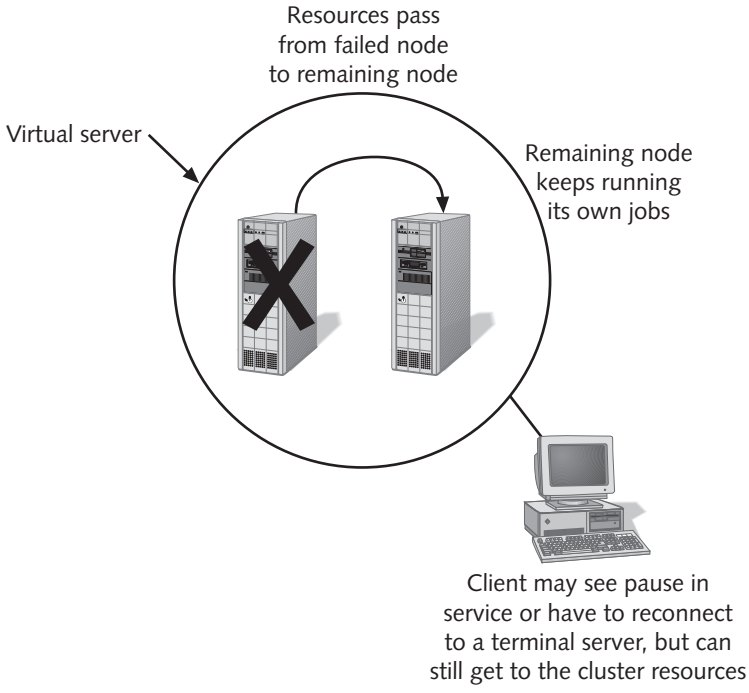

Figure 13-5 How active/active clusters fail over

In an **active/standby cluster** (see Figure 13-6), only the primary node actually handles client requests. The secondary node waits for the main server to fail (it may be doing something else, but it is always "on call" in case the primary node cannot fulfill a client request). If the active node fails, it passes its workload to the standby node. Failover time in this case takes 15 to 90 seconds. If the standby node was doing something before the failover, all of those operations stop and only the operations that had been running on the failed server are carried out.

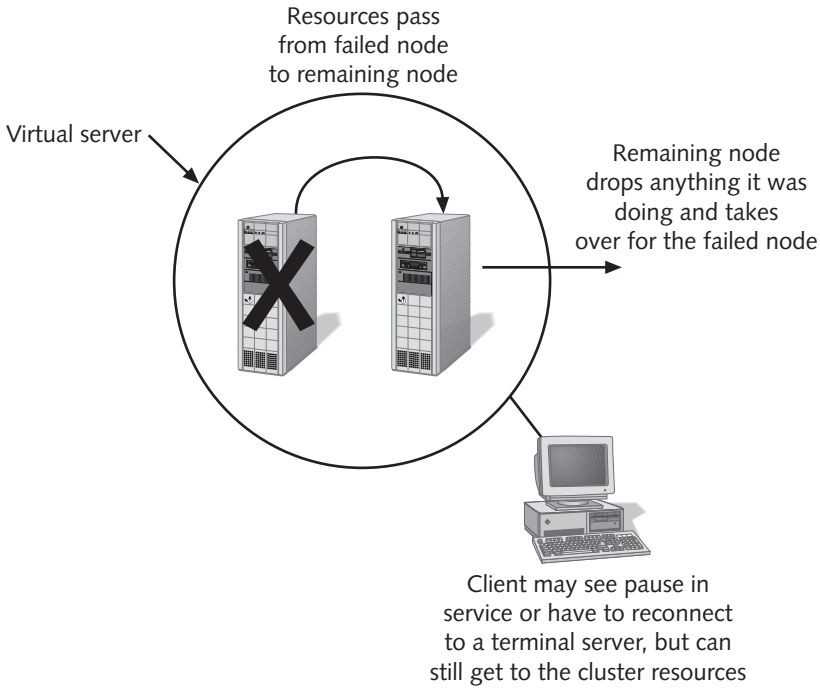

Figure 13-6 How active/standby clusters fail over

In these cluster types, the nodes in the cluster don't have to be doing the same thing. In an active/active cluster, the secondary node that takes over for the failed primary server adds the failed node's jobs to its own. In an active/standby cluster, the secondary node drops what it was doing and replaces its workload with that of the failed primary node. In neither case does the node hardware have to be identical. That is, the two devices must be identical in function—capable of storing the same data in the same place or running the same applications (clustered e-mail servers obviously must have e-mail server software installed)—but they don't have to be physically identical.

In both the active/active and active/standby cluster types, any client connections to the failed primary node are terminated when the server fails. Any services running on the primary server also stop.

In a **fault-tolerant cluster**, the setup is a little different. In this case, all of the nodes in the cluster are identical and operate in tandem, so all nodes in the cluster are always performing the same operations all the time. If one node fails, another node can take over for it nearly instantaneously, with less than one second required for the failover. Like disk mirroring, fault-tolerant clustering is inefficient in space because it provides 100% redundancy of data and function; like mirroring, it also is an excellent way of keeping data available with no downtime. Windows 2000 clustering is not really fault-tolerant in this sense.

The physical connection may also vary depending on the clustering package implemented. Today, clustered servers typically communicate via a high-speed SCSI connection, or via a fiber channel fabric that provides network connectivity at local-access speeds.

How Clustering Works

To create a new cluster, you run a cluster installation tool on a server that will be part of the new cluster and name the new cluster. This server becomes the first node of the new cluster. You can then use the administration tools to configure the parts of the server that should become resources.

To add more nodes to the cluster, you run the installation service on each server in turn, filling in the name of the cluster (it is like a workgroup, in a way) when prompted. As each node joins the cluster, the cluster service of the node manager on the first node sends the new node a copy of the existing cluster database that is maintained by the configuration database manager.

Thereafter, whenever a node in the cluster boots up, the cluster service starts automatically and broadcasts messages to find the cluster that it joined earlier. If the cluster answers and the node supplies the correct password to join the cluster, it is in. If the node can't find its cluster or its password is refused, the node attempts to form its own cluster. To form a resource, it must get access to a quorum resource.

When you power down a node, it sends a message to the other cluster members that it is leaving the cluster. Although the node doesn't wait for any acknowledgment of its departure, this message means that all other nodes in the cluster are immediately aware of the changed membership of the cluster (maintained by the node manager) and don't have to go through the discovery process again to discover that the node is gone.

Client–Cluster Communications

As mentioned earlier, the nodes in a Windows 2000 cluster are exposed to the client as a single virtual server. This server is identified to the outside world with a single network name and network IP address. (Windows 2000 clustering requires the use of TCP/IP as the network protocol; IPX/SPX and NetBEUI don't work with clustering.)

Each node in the cluster is set up to perform the same job and share a common disk array; therefore, if one server fails or an application stops working, the cluster service can fail over the resources from the failed node, and another node can take its place. If a node or an application on a node stops responding, the clustering software restarts the application on another server in the cluster or redistributes the work of the failed server to another server in the cluster. The person using the server should see only a slight pause in service during the failover. For example, you might cluster two e-mail servers. Then, even if one server fails, network users can still pick up their e-mail. Clustering the e-mail servers could also provide high availability by **load balancing**, or distributing the client requests among the nodes in the cluster. Because load balancing always gives new jobs to the least busy node, it improves availability by reducing server response time.

Although nodes in a cluster can be load balanced, not all load-balanced servers are clustered. For example, software can load-balance identical terminal servers so that the least busy terminal server always services a new client request. These load-balanced servers are not necessarily clustered, however. In fact, it is rare that terminal servers are clustered, because the

added cost of clustering is greater than the benefits derived from making the servers fault-tolerant. If a terminal server fails, any users running terminal sessions must reconnect to another terminal server to continue working, and they lose any data still in RAM and not saved to a file server. That said, an active/active cluster (like the one supported by the Windows 2000 cluster service) would lose this information anyway. Although most resources can be restarted gracefully after failover, any current operations will be lost. For example, any current print jobs will be aborted and must be restarted, and any database queries will be lost and must be reissued.

Requirements for Clustered Applications

Not all applications work in a clustered environment. In fact, many don't use clustering at all. For an application to support clustering, all of the following must be true:

- The application must use TCP/IP for communications, not NetBEUI or IPX/SPX.
- The client-side applications and operating system must not crash or hang if a cluster must fail over a client to another client. If a server or application on a server fails, the client must be able to automatically or manually retry the connection to the virtual server until the failover is complete and the other server can take the connection.
- You must be able to configure server applications so that they store only data on the disk array common to all servers in the cluster.
- Applications should not store large data structures in the Registry. The cluster service logs any changes to the Registry in servers in the physical cluster. When a server application fails over to another node, the cluster service checks the log for changes to the Registry on the failed server and then replicates those changes to the secondary server; hence the same settings become available to the application when it restarts on the secondary server. Putting large amounts of data in the system Registry stresses the network by increasing the amount of data that must be replicated, thereby increasing failover time.
- The applications must be able to restart on the secondary server even if the primary server fails suddenly.
- Applications must refer to the virtual server name and IP address, not the server name and IP address of a physical computer. Otherwise, when the application fails over to the secondary server, the application won't work because it won't be able to find the physical server on which it was installed.
- A server application must be installed on all servers in the cluster. Thus, if one sever fails, the secondary server can take over. The server application must be configurable as a **cluster resource**, an object that can be moved between servers in the cluster. The cluster service then starts and stops the application by taking the cluster resource representing the application online and offline.

Chapter Summary

- Fault tolerance is a matter of keeping available the tools you need. To keep data available, you can (and should) use backups. To keep data available without having to restore data to a server, you can use fault-tolerant disk configurations. Finally, to keep data available even if an entire server goes down, you can cluster servers. In each case, the name of the game is redundancy—whether redundancy of data, redundancy of disk space, or redundancy of servers.
- Sometimes, even fault tolerance can't do everything. That situation is addressed in Chapter 14, which details the Windows 2000 disaster recovery tools.

Key Terms

active/active cluster — A type of cluster found in Windows 2000 Advanced Server and Windows 2000 Datacenter in which both nodes in the cluster are serving client requests all the time. If one node fails, the cluster service moves its resources to the other node and the second node manages both workloads. The two nodes do not have to be identical.

active/standby cluster — A type of cluster in which one node is serving client requests and the other is dormant, or doing work that may be discarded. When the active node fails, the cluster service moves its resources to the standby node, and the standby node drops whatever it had been doing and takes over the active server's workload. The two nodes do not have to be physically identical.

archive bit — An attribute that is set (turned on) when a file is created or edited, and turned off when the file is backed up.

bit mask — The result of an XOR function. The bit mask contains a 1 for every mismatch between numbers and a 0 for every match.

breaking the mirror set — Reverting both halves of a mirror set to independent simple volumes.

clustering — A technique that involves logically combining two or more servers for redundancy.

cluster resource — An object that can be moved between servers in the cluster.

cluster service — A Windows 2000 service running on all nodes of a cluster, facilitating communication and failover between nodes.

copy backup — A type of backup that works like a normal backup in that it copies all selected files to the backup media, regardless of whether the archive bit is set. A copy backup does not reset the archive bit.

daily backup — A type of backup that copies only those files that have changed on the day of the backup.

deleting the mirror set — Removing the mirror set volume and thus discarding all of its data.

differential backup — A type of backup that copies to the backup media every selected file that has the archive bit set, but does not reset the archive bit.

disk duplexing — A mirror set that incorporates two disks attached to different disk controllers, so they are not affected by controller failure.

disk mirroring — A RAID type that combines space on two physical disks to create a mirror image; that is, when data are written to one disk, the same information is written to the other disk.

disk striping with parity — See *RAID 5 volume*.

exclusive OR (XOR) arithmetic — The function that RAID 5 volumes use to calculate the parity information for their data. When calculating the XOR for two binary numbers, you compare them side by side. The result will have a 0 in every place where the numbers match and a 1 in every place where they do not.

fault tolerance — An aspect of an operating system that ensures high availability of both user and system data and of the computing resources.

fault-tolerant cluster — A cluster type in which two physically identical nodes operate in tandem, performing the same functions. If one node fails, the other takes over for it almost instantly because no resource transfer is needed.

Grandfather/Father/Son (GFS) — A backup scheme that uses a monthly normal backup, a weekly normal backup, and a daily incremental or differential backup to create a three-month record of server data.

group — A collection of like cluster resources that the cluster service can manage as a single logical unit.

incremental backup — A type of backup that copies to the backup media every selected file that has the archive bit set and then resets the archive bit to show that the file has been backed up.

load balancing — Distributing client requests among grouped (but not necessarily clustered) servers so that the least busy server always services the next client request.

mirror set — The name for the combined disk space that is turned into a disk mirror.

nodes — Individual servers in a cluster.

normal backup — A type of backup that copies every selected file to the backup media and resets the archive bit on the original files. This backup type is the core of a backup strategy.

NTBACKUP — The backup program that comes with Windows 2000. It is accessed by selecting Start, Run, and typing NTBACKUP.

online — In terms of fault tolerance, when a resource provides its service on its node.

quorum resource — A cluster resource that is used as a tiebreaker when two servers are trying to form a cluster at once. The one with control of the quorum resource controls the cluster.

RAID 5 volume — An elaboration of disk striping in which parity information for the data written to the volume is also written to the volume. If one disk in a RAID 5 volume fails, the data that it contained may be reconstructed from the parity information on the remaining disks.

Redundant Array of Independent Disks (RAID) — The technique of logically combining physical disks to make fault-tolerant disk volumes. If one disk in a RAID array fails, the other disk or disks can take over until the broken disk can be replaced.

removing the mirror set — Discarding one half of a mirror set's data (converting the volume to unallocated space) and reverting the other half to a simple volume.

resources — Part of a cluster (hardware or software) that the cluster software manages. The cluster service includes DLLs that represent some common potential resources, and developers can build their own.

security identification (SID) number — A unique number assigned by Windows 2000 to each user account.

selection information file — The file in which backups are stored. This file has a .bks extension.

system state data — The Windows 2000 name for system configuration information. System state data include the Registry, the boot files, the class registration database, and, if applicable, the certificate services database, Active Directory structure, and SYSVOL.

Tower of Hanoi (ToH) — A backup scheme that uses five tapes in rotation to create a 32-week record of normal backups. Because this backup scheme does not include differential or incremental backups, it should not be used as the sole backup plan.

virtual server — The name by which the nodes in a cluster are collectively known. Clients connect to the virtual server, not to the individual nodes within the server.

Review Questions

1. Which of the following is *not* a resource for which the Windows 2000 cluster server already has a DLL?
   a. Print shares
   b. Applications
   c. Databases
   d. None of the above
2. One of the disks in your mirror set has stopped working. To make the data in the mirror set fault-tolerant again, what do you need to do first?
   a. Back up the volume.
   b. Delete the mirror set.
   c. Break the mirror set.
   d. Remove the mirror set.
3. What is the minimum number of disks that you need to make a RAID 5 volume? A mirror set?
4. The _____ in a cluster are logically combined to create a virtual server.
5. Briefly describe the role of a quorum resource in a cluster.
6. Which node(s) in a cluster does the cluster service run on?
   a. The lead node
   b. The node controlling the quorum resource

c. All nodes in the cluster

d. The node maintaining the cluster configuration database

7. Which of the following is *not* part of the system state data for a Windows 2000 member server?

   a. The COM+ class registration database of file associations

   b. The boot files for the computer

   c. The Active Directory structure

   d. None of the above

8. List the parts of the system state data that are part of a domain controller but not part of a member server.

9. You've set up a RAID 5 volume on five disks ranging in size from 100 MB to 5 GB. Each partition in the volume is 100 MB in size. How much of the RAID 5 volume will be used to store user data and how much will be used to store parity information?

   a. 100 MB

   b. 50 MB

   c. 500 MB

   d. 250 MB

10. You have two 2 GB dynamic disks that you want to support a mirror set. One of the disks has 250 MB of unallocated space; the other has 450 MB of unallocated space. How large a mirror set could you create from these two disks?

11. Taking the same disk configuration in question 10, assume that you made the largest mirror set possible. If the mirror set was full, how much disk space would be taken up with redundant data? How much with user data?

12. The backup tool in Windows 2000 is located in the Administrative Tools folder. True or False?

13. What is a selection information file?

    a. The file in which Windows 2000 backup jobs are stored

    b. The file on a disk that tells you which partitions are fault-tolerant

    c. A record of changes to the volume structure of an NTFS partition

    d. The record of which nodes are in a cluster

14. To back up data with Windows 2000, you need a tape drive. True or False?

15. You want to automatically generate a complete list of all files that were backed up during a single backup procedure. How would you accomplish this task?

16. Which of the following are fault-tolerant disk configurations that Windows 2000 supports in software? (Choose all that apply.)

    a. Disk striping

    b. RAID 5 volumes

    c. Mirror sets

    d. All of the above

17. How does the logical disk manager fit into Windows 2000 fault tolerance?
18. Which backup media can use hardware compression?
    a. Removable disks
    b. Floppy disks
    c. Tape drives
    d. All of the above
19. Which fault-tolerant disk configuration would you want to use to protect a compute-bound server?
20. Which backup type will back up only those files with the archive bit set and then reset the archive bit?
    a. Normal
    b. Differential
    c. Incremental
    d. Copy
21. It is simpler to completely restore a server's data from a week's worth of differential backups than from a week's worth of incremental backups. True or False?
22. Briefly explain how normal backups and copy backups differ.
23. You want to protect the data on your server, but need to keep initial costs as low as possible. Which Windows 2000 fault-tolerant disk configuration would you choose, and why?
24. What does 10001011 XOR 10101001 equal?
    a. 00100010
    b. 10101011
    c. 00110101
    d. 00101011
25. You want to take the data in a mirror set and insert it in a new computer without affecting the original data in any way. Before you remove the disk and put it in the new computer, what do you need to do to the mirror set?
    a. Break it.
    b. Delete it.
    c. Remove it.
    d. Copy it.

HANDS-ON PROJECTS

Project 13-1

To create a mirror set and format it with FAT32:

1. Start the Computer Management tool by selecting **Start**, **Programs**, **Administrative Tools**, **Computer Management**.
2. Expand the **Storage** section and select the **Disk Management** tool in the right pane. Make sure that at least two dynamic disks with unallocated space are available.
3. In the right pane, right-click an area of unallocated space on one of the dynamic disks, or right-click the gray disk label to the left. From the context menu that appears, choose **Add Volume** to start the Create Volume Wizard.
4. Click past the opening screen, then choose **Mirrored Volume** in the **Select Volume Type** page (see Figure 13-7). Click **Next**.

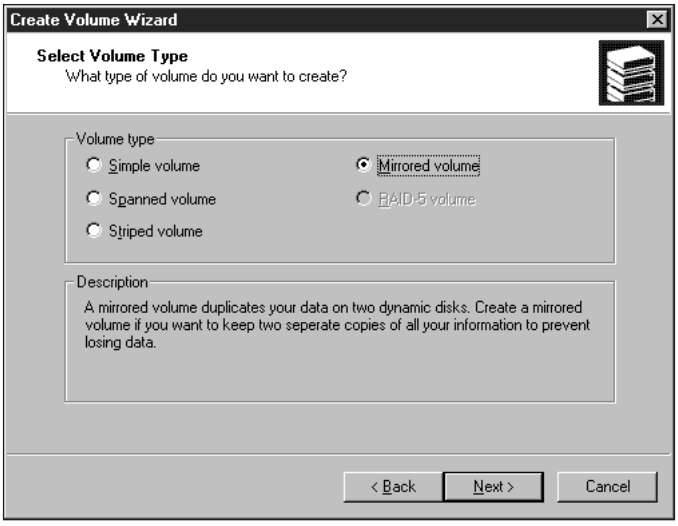

Figure 13-7 Choose to create a mirror set

5. From the Select Disks screen shown in Figure 13-8, choose the disks that you want to be in the mirror set. The disk from which you started the process will already be selected. Choose another disk from the ones listed in the column on the left.

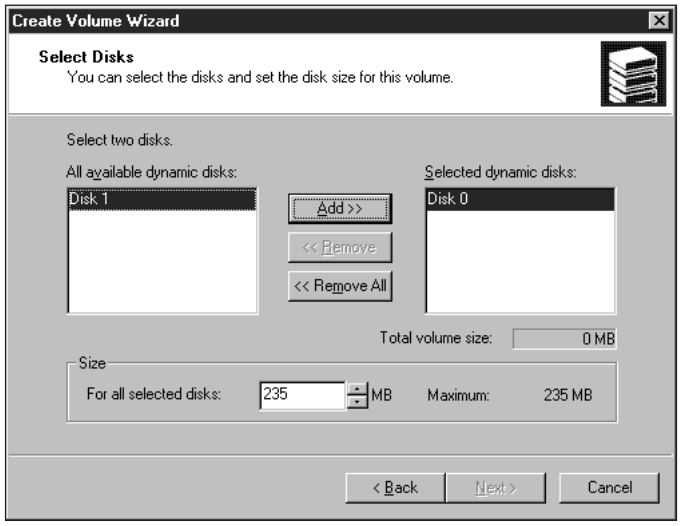

Figure 13-8 Pick the disks that you want to form the mirror set

If you have three or more dynamic disks available from which to create the mirror set, you can deselect the current disk and select two others from the list of available disks.

6. While still in the Select Disks screen, type a size for the mirror set, ranging from 1 MB to the maximum possible size. The size you choose should reflect the total storage area available. Click **Next**.

Because the two areas of unallocated space must be the same size, the maximum possible size for the mirror set depends on the amount of unallocated space on which you build the disks. Pick and choose the disks you want to include to see how the maximum size of the mirror set changes accordingly.

7. In the Assign Drive Letter or Path screen shown in Figure 13-9, assign a drive letter to the mirror set or mount it to an empty volume on an NTFS partition as described in Chapter 7, and then click **Next**.

You create a partition with the Disk Management tool, which is accessed by selecting Start, Programs, Administrative Tools, Computer Management. From within Disk Management, select to create a primary partition.

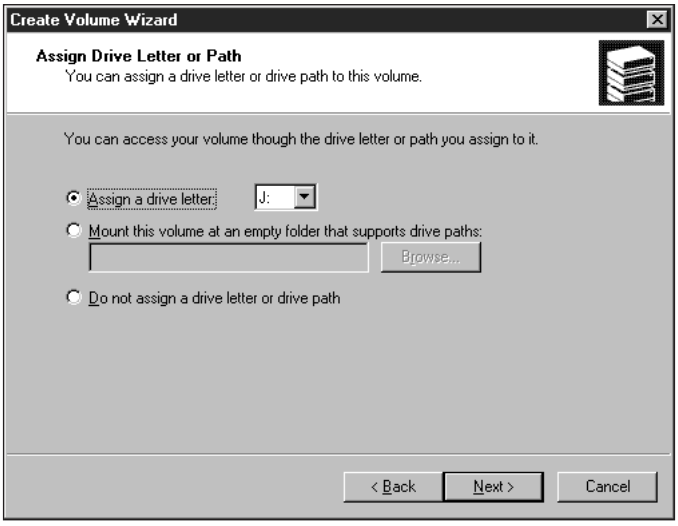

Figure 13-9 Label the volume in some way so that you can access it

8. In the Format Volume screen (see Figure 13-10), choose **FAT32** from the list of available disk formats. Click **Next** when you are done.

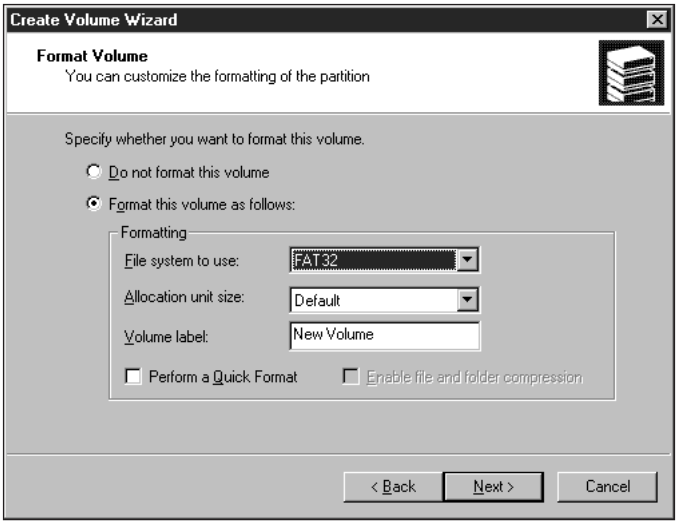

Figure 13-10 Format the new mirror volume with FAT32

9. Review your choices, and click **Finish** to start building the mirror set. The Disk Management tool chugs away for a few minutes to create the volumes and set them up to communicate. The mirror set will be available right away—you don't need to reboot.

Project 13-2

To mirror an existing simple volume:

1. Start the Computer Management tool by selecting **Start**, **Programs**, **Administrative Tools**, **Computer Management**.

Only dynamic disks with areas of unallocated space large enough to mirror the selected volume are listed. If no area of unallocated space is large enough, you won't have the option of mirroring the volume.

2. Right-click the volume you want to mirror, and then choose **Add Mirror** from the resulting menu. An Add Mirror dialog box similar to the one shown in Figure 13-11 appears, asking you to select the disk on which you want to create the mirror. Click the disk so that it is highlighted—this operation won't work otherwise.

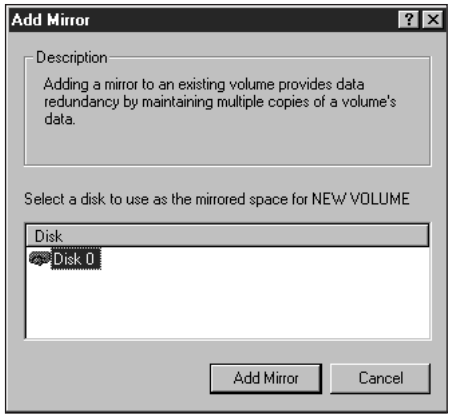

Figure 13-11 Choose a dynamic disk with unallocated space on which to mirror the simple volume

3. Click the **Add Mirror** button. In the unallocated space, the Disk Management tool will create a partition that is the same size and format as the simple volume that you are mirroring. The redundant data are regenerated onto the new partition. (Depending on the size of the volume you're mirroring, this step may take a while. It is not a fast process on large volumes.)
4. The new partition will have the same drive letter or mounted path as the one you mirrored and will be available as soon as all of the data are regenerated—no reboot is required.

Project 13-3

To use the Windows 2000 graphical backup tool to create an incremental backup job that backs up the contents of the My Documents folder to a network location and then verifies the data after it's backed up:

1. Choose **Start**, **Programs**, **Accessories**, **System Tools**, **Backup**.
2. Click the **Backup Wizard** button on the Backup utility (see Figure 13-12).

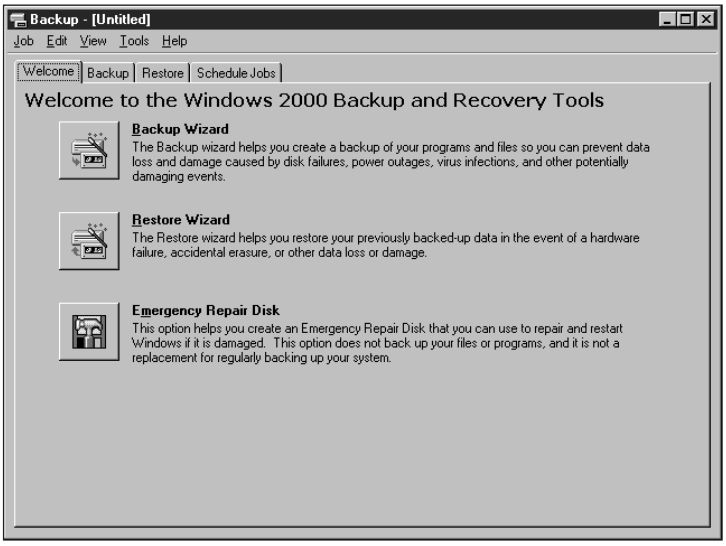

Figure 13-12 Opening screen of the Backup tool

3. Click past the opening screen of the wizard, and in theWhat to Back Up screen choose **Back up Selected files, drives, or network data**. Click **Next**.
4. In the Items to Back Up screen, choose **My Documents**. When the folder is selected, it will have a blue check mark in the box next to the folder's icon. Click **Next**.
5. In the Where to Store the Backup screen shown in Figure 13-13, provide a name and location for the backup's selection information file. Click **Next**.

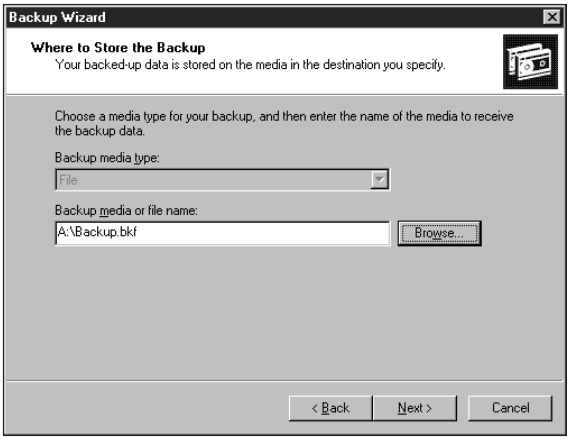

Figure 13-13 Choose a destination location for the backup

6. Review the options displayed in the Completing the Backup Wizard screen. In this example, the Verify setting is currently off and it is a normal backup. To turn the Verify setting on, you'll need to use the Advanced options. Click the **Advanced** button.
7. In the Type of Backup screen shown in Figure 13-14, choose **Incremental** from the drop-down list. Click **Next**.

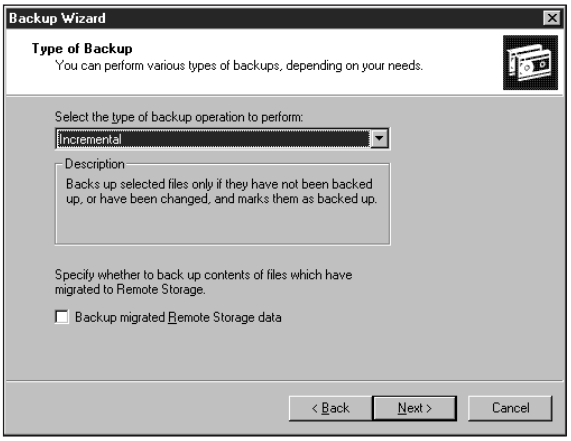

Figure 13-14 Choose to create an incremental backup

8. In the How to Back Up screen of the wizard, choose **Verify data after backup**. Click **Next**.
9. Review the rest of the options in the Advanced Backup Wizard, noting that you can choose to append or replace data, specify a new backup label and job name (perhaps "Incremental copy of the My Documents folder"), and choose to schedule the backup for a later time. When you reach the end of the wizard, click the **Finish** button to start the backup.

Project 13-4

To create a new RAID 5 volume and mount it to an empty new folder on a NTFS volume:

1. Make sure that at least three dynamic disks are installed in the server and start the Computer Management tool by selecting **Start**, **Programs**, **Administrative Tools**, **Computer Management**.
2. Right-click any area of unallocated space on any dynamic physical disk. From the resulting menu, choose **Create Volume**.
3. Click **Next** in the opening screen of the Create Volume Wizard. On the Select Volume Type page shown in earlier in Figure 13-7, select **RAID-5 Volume** and then click **Next**.
4. In the Select Disks screen shown in Figure 13-15, choose at least three disks that you want to be involved in the stripe set. The disk with which you started (the one with the area of unallocated space) will appear in the right column of disks to use; the other dynamic disks with unallocated space will appear on the left side. To remove a disk from the volume, select it in the list of selected dynamic disks and click the **Remove** button.

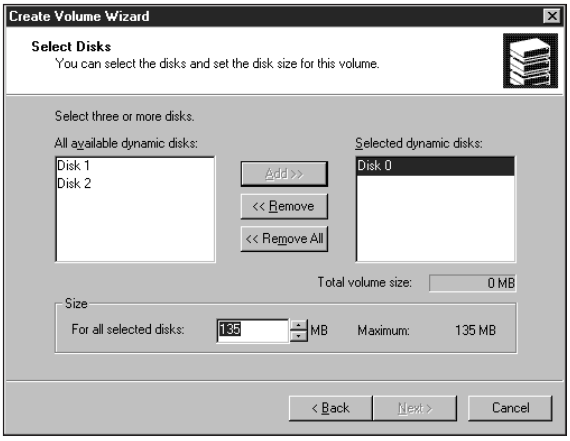

Figure 13-15 Pick the disks that you want to be part of the RAID 5 volume and choose a size for the volume

5. In this same screen, pick the size of the stripe set. In the Size box, the Disk Management tool displays the maximum size of the stripes based on the unallocated space on the chosen drives. The value in Total Volume Size reflects the total amount of room available for data, not the total space in the stripe set. Click **Next**.

Because 1/*n* of the space in a RAID 5 volume (where *n* is the number of disks in the set) is used for parity information, the more disks you have, the larger percentage of data storage the volume will have.

6. In the Assign Drive Letter or Path screen, choose **Mount This Volume** and click the **Browse** button.
7. In the Browse for Drive Path dialog box that appears (see Figure 13-16), you'll see all of the logical drives formatted with NTFS. Select one (in our example, we chose D) and click the **New Folder** button. Type a name for the new folder in the space provided, and then click **OK** to return to the original screen.

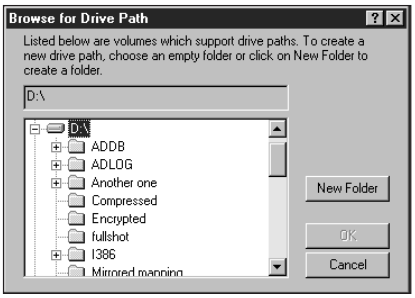

Figure 13-16 Find the local NTFS volume to which you want to mount the RAID 5 volume

8. In the Format Volume screen, choose whether to format the new volume immediately and, if so, the disk format you want to use (NTFS, FAT32, or FAT).
9. Review your choices, backing up to change any of them or clicking **Finish** to create the RAID 5 volume. Windows 2000 will grind away for a few minutes, setting up the new stripe set. When it's done, the RAID 5 volume will be immediately ready to use—no reboot is required.

Project 13-5

To use the command-line NTBACKUP utility to perform a differential backup of C:\MYFILES and label it accordingly:

1. Choose **Start**, **Programs**, **Accessories**, **Command Prompt**.
2. In the Command Prompt window, type **ntbackup /?** to get a complete list of all NTBACKUP switches that control the backup options. It is a long list, but the options you'll be looking at for this exercise are the /m, /f, and /j switches.
3. Type the following command and then press **Enter**:

   ntbackup c:\myfiles\ /m differential /f d:\system1.bkf /j "Differential backup of MYFILES folder"

Project 13-6

To break the mirror set you created earlier so that the two halves are separate but retain their data:

1. Start the Computer Management tool by selecting **Start**, **Programs**, **Administrative Tools**, **Computer Management**.

2. Right-click a mirror set and choose **Break Mirror** from the menu that appears. You'll see a dialog box like the one in Figure 13-17, warning you that breaking the mirror set will make it no longer fault-tolerant.

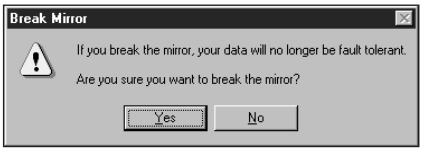

Figure 13-17 Confirm that you want to break the mirror

3. Click **Yes**, and the two halves will become separate simple volumes. One half will retain the drive letter that you assigned the mirror set, and the other will take the next drive letter available. Both halves will retain any volume label that the mirror set had.

Case Projects

1. You have four disks in a server. To retain dual-boot capabilities with an alternative operating system, you want to keep one of them a basic disk. Which fault-tolerant disk configurations are available if you use only the Windows 2000 RAID software? Which option would you choose if you were most concerned about the speed of disk writes? What if you were concerned about getting as much storage space out of the fault-tolerant array as possible?
2. Which Windows 2000 backup types would you use to back up a server so that you always have a daily record, can restore the backups in as few steps as possible, and have an archive of old backups?